Title: X Clone – A Firebase-Powered Social Media Web Application

**Introduction**

In today’s world, social media platforms play a vital role in information sharing, networking, and public expression. Twitter (now rebranded as X) stands as a real-time microblogging platform that allows users to express thoughts in short posts called tweets.

Our project aims to build a functional clone of X (Twitter) using React for the frontend and Firebase for the backend. The application features a clean dark mode UI and core functionality like posting tweets, viewing a live feed, logging in securely, and navigating with a sidebar — providing users with a realistic experience.

**Objectives**

* To replicate Twitter’s core features in a scalable and modular way.
* To use Firebase Authentication for secure user sign-in/sign-up.
* To integrate Firestore for storing and retrieving tweets in real time.
* To build a visually pleasing and user-friendly dark-themed UI using React.
* To demonstrate core full-stack development concepts.

**Problem Statement**

Building a social media application from scratch is complex, involving secure authentication, real-time data handling, and an intuitive UI. This project demonstrates how such complexity can be tackled using React and Firebase, providing engineering students with real-world app development experience.

**Scope of the Project**

* Authenticated users can log in and post tweets.
* Tweets are instantly visible to all users through a live feed.
* Modular design allows easy scalability and addition of features like comments or likes in the future.
* The project operates locally and is intended for demonstration/educational purposes.

**Tools & Technologies Used**

| **Technology/Tool** | **Purpose** |
| --- | --- |
| React.js | Frontend library for building UI |
| Firebase | Backend-as-a-Service platform |
| Firebase Auth | For email/password-based login system |
| Firestore Database | Cloud NoSQL database for storing tweets |
| CSS Modules / CSS | Styling with scoped classes for dark theme |
| Visual Studio Code | Code editing environment |
| ESLint & Prettier | Optional - for code formatting and linting |
| Node.js + npm | Package management and running local server |

**System Architecture**

The app uses client-side architecture with Firebase for backend services:

Frontend:

* React handles component rendering and routing.
* Components: Sidebar, Feed, Tweetbox, Widgets, Auth.

Backend:

* Firebase Auth manages user login and session handling.
* Firestore stores and retrieves tweets in real time.

**Features Implemented**

* User Authentication (Sign in / Sign up).
* Reusable and modular component design.
* Posting tweets and rendering in feed.
* Dark mode UI.
* Real-time data fetching using Firestore.
* Responsive layout suitable for desktops and mobiles.
* Minimal, distraction-free design.

**Firebase Configuration**

Firebase was initialized using the following steps:

1. Created a Firebase project in Firebase Console.
2. Enabled Email/Password authentication.
3. Set up Firestore database in test mode.
4. Added Firebase SDK to project via npm.
5. Initialized Firebase in firebase.js and exported auth, db.

firebase.js:

import { initializeApp } from 'firebase/app';

import { getAuth } from 'firebase/auth';

import { getFirestore } from 'firebase/firestore';

const firebaseConfig = {

  apiKey: "AIzaSyAdrfPLpD3edLKicWd\_xFxuGIXwxzgLen4",

  authDomain: "x-clone-b096d.firebaseapp.com",

  projectId: "x-clone-b096d",

  storageBucket: "x-clone-b096d.firebasestorage.app",

  messagingSenderId: "441640319388",

  appId: "1:441640319388:web:2dc2137b0778b78484f059"

};

const app = initializeApp(firebaseConfig);

const auth = getAuth(app);

const db = getFirestore(app);

export { auth, db };

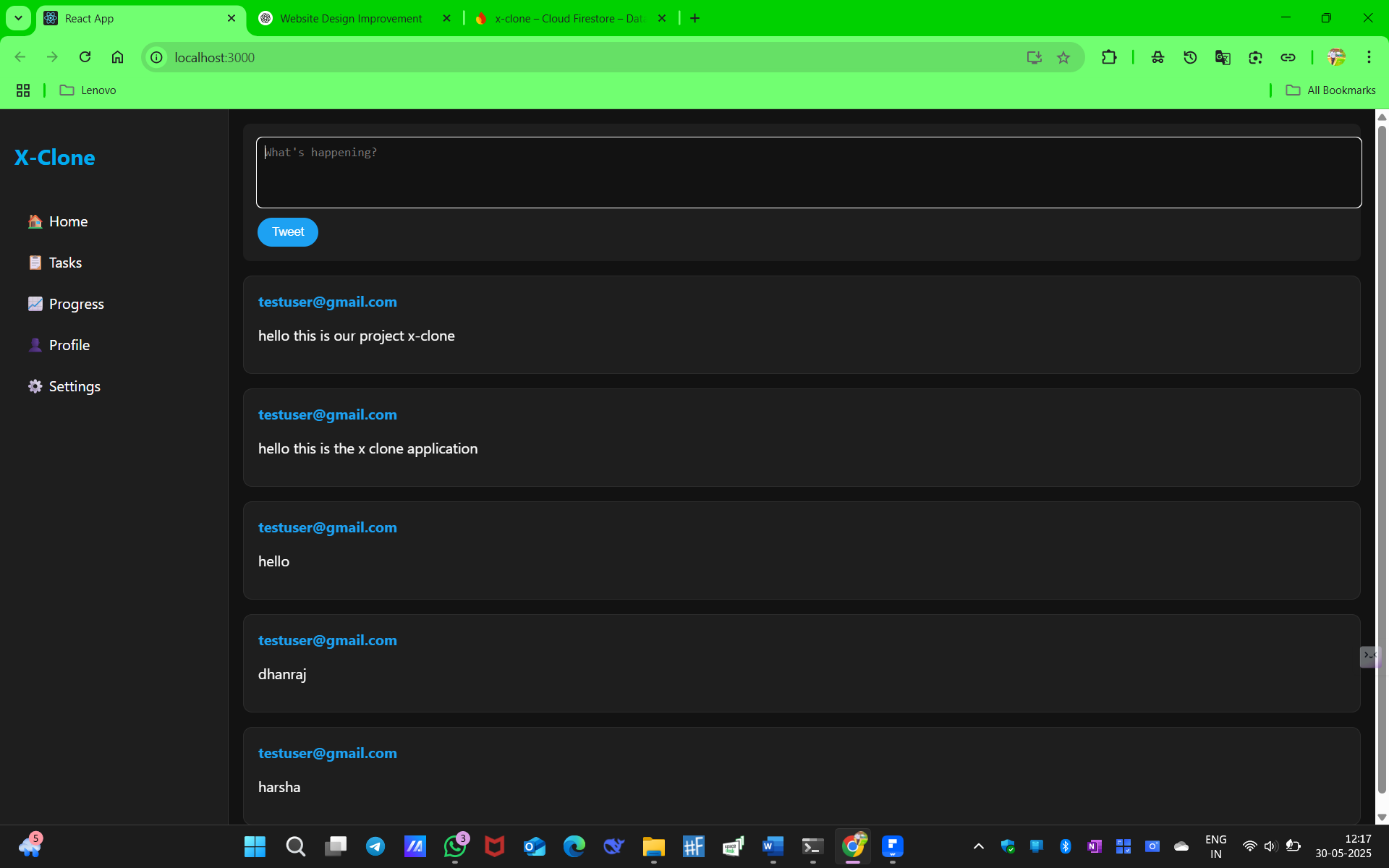
**Methodology**

We followed an iterative development methodology:

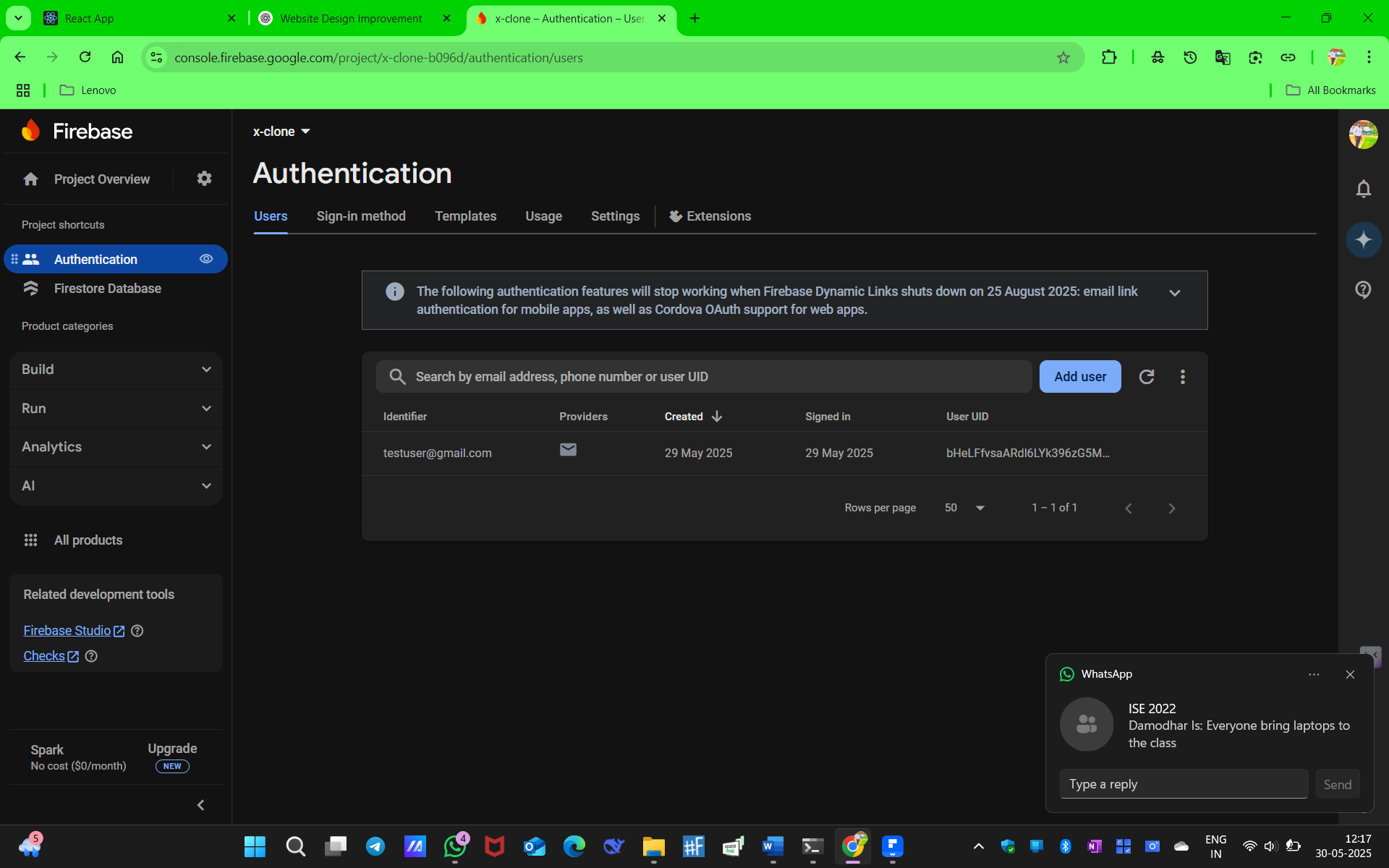
| **Phase** | **Description** |
| --- | --- |
| Planning | Requirement analysis and deciding tech stack |
| Design | UI design and component breakdown |
| Development | Coding using React and Firebase integration |
| Testing | Checking component logic and Firebase workflows |
| Documentation | Final report and screenshots added |

**Screenshots**

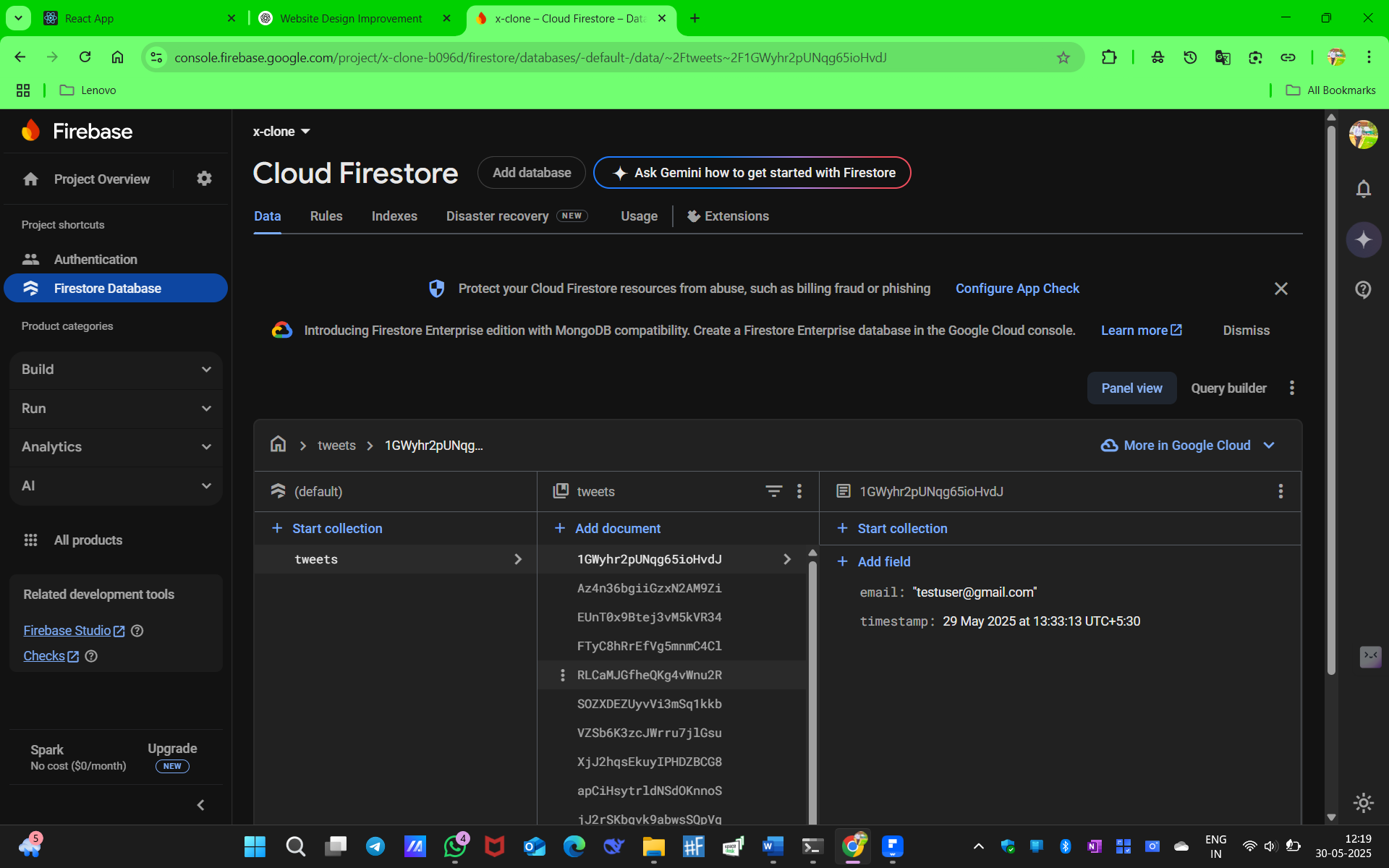
* Tweet Posting Section (Tweetbox)



* Firebase Authentication Dashboard



* Firestore Console (Tweet Collection)



**Future Enhancements**

* Add profile pages and individual user timelines.
* Enable media/image uploads in tweets using Firebase Storage.
* Implement likes, retweets, and comment sections.
* Add notifications and tweet analytics.
* Deploy on Firebase Hosting or Netlify.

**Conclusion**

This project gave us hands-on experience in full-stack web development. It successfully mimics the core functionality of Twitter/X using Firebase and React. We were able to build a secure, modular, and real-time web app that serves as a strong foundation for more advanced applications. It not only enhanced our coding skills but also taught us the importance of scalable design and real-time databases.